

# Databases and SQL

Evi Berchtold

27.06.2011

## 1 Introduction

The main purpose of databases is to store data consistently, permanently and efficiently. A database system consist of the database itself and a database management systems (DBMS) which provides methods to handle databases efficiently [1]. There are several different ways to organize the data. For example there are hierarchical or network based models. But the most commonly used model for databases is the relational database that was first proposed by E.F. Codd in 1969 in his paper "A Relational Model of Data for Large Shared Data Banks". In this model the data is stored in tables that are also called relations. The operations to modify the data are based on relational algebra, so in contrast to other database models it can be described in formal mathematical terms [2].

One of the first commercial languages for the relational model was developed at IBM by Donald D. Chamberlin and Raymond F. Boyce. Its name SQL is short for Structured Query Language. Even though it deviates from the original relational model in many details, it became the most widely used database language [3].

Especially in web applications MySQL is a popular choice for a relational database systems. It is used by many famous web sites as for example YouTube, Wikipedia, Google or Facebook. It is distributed as open source under the GNU General Public License (GPL) but there are also commercial licences that offer additional support [4].

This week's programming challange is to set up a MySQL database server, create a database and a user and manage the permissions of this database. Furthermore, a programming language like perl should be used to create a table, a backup should be created and *phpadmin*, a web-based user interface should be configured.

## 2 MySQL installation

First, we have to install the package *mysql-server* by the command

```
apt-get install mysql-server
```

Aptitude automatically also installs several other packages which are needed like *mysql-client-5.1*, *mysql-server-5.1* or *libdbi-perl*. During the installation process the user is asked to set a root password. It is important to set a root password as otherwise everyone could login as root and have all the permission to e.g. delete the whole database.

The default configuration file is located in */etc/mysql/my.cnf*. In this file several settings of the MySQL server can be defined, as e.g. the TCP/IP port (default=3306), the number of allowed concurrent sessions (default=151) and the size of the query cache (default=1MB). The command `SHOW VARIABLES` will show all current values for the server system variables that can be defined in the configuration file. As we are running our MySQL server on a virtual image with limited RAM, the default configuration should be replaced with a configuration file that is designed specifically for this situation and can be found in */usr/share/doc/mysql-server-5.1/examples/my-small.cnf* [5].

## 3 MySQL administration

MySQL has a special database called *mysql* to manage users and permissions [5]. In order to create a new database and a new user we have to add several entries to the tables in this database. So we connect to the database as root and change the database to *mysql*:

```
mysql -u root -p
use mysql
```

In order to give a user specific rights three tables have to be modified: *host*, *user* and *db*. These tables consist of so-called scope and privilege columns. Scope columns define in which context the privileges given in the privilege columns apply.

The *user* table defines whether connections of a user from a host are allowed. A user is always identified by its user and host name, as it is possible that two users on different hosts happen to have the same user name. So to have a user "bercht" with password "XXX" that can login from "localhost" and "bercht.course" we insert the following two entries into user:

```
mysql> insert into
-> user (host, user, password)
-> values ('localhost', 'bercht', password('XXX'));
mysql> insert into
-> user (host, user, password)
-> values ('bercht.course', 'bercht', password('XXX'));
```

The `password()` function should always be used to encrypt the password, as it would otherwise be stored in the database in clear text.

The *host* table manages together with the *db* table the rights of the users on the databases depending on the host from which they login. If we do not want to this for each database separately, we can use `'%'` as a wildcard for all databases. We want to grant users from "localhost" and "bercht.course" permissions for select, insert, update, delete, create and drop statements on all databases:

```
mysql> insert into host
-> (host, db, Select_priv, Insert_priv, Update_priv,
-> Delete_priv, Create_priv, Drop_priv)
-> values('localhost', '%', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y');
mysql> insert into host
-> (host, db, Select_priv, Insert_priv, Update_priv,
-> Delete_priv, Create_priv, Drop_priv)
-> values('bercht.course', '%', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y');
```

In the *db* table it is defined which rights a user-host combination has on a specific database. If we want a user to be able to login from various hosts, we can leave the host value in *db* empty. In this case the hosts specified in the *host* table will be used.

```
mysql> insert into
-> db (host, db, user, Select_priv, Insert_priv, Update_priv,
-> Delete_priv, Create_priv, Drop_priv)
-> values (NULL, 'bercht_db', 'bercht', 'Y', 'Y', 'Y',
-> 'Y', 'Y', 'Y');
```

Normally the more simple commands `CREATE USER`, `GRANT` and `REVOKE` are used to manage user permissions, but they do not change the *host* table. So in order to let a user login from various hosts at least the *host* table would have to be manipulated manually nevertheless.

Finally we have to create the database "bercht\_db" and reload the MySQL server:

```
mysqladmin -u root -p create bercht_db
mysqladmin -u -p reload
```

Afterwards we should be able to connect to the database as user "bercht" and use the database "bercht\_db".

## 4 MySQL Perl API

For many programming languages there exists an API for MySQL database access. Part of this week's programming challenge is to write a perl script to create a table in our database. Therefore, perl and the libdbi-perl package have to be installed. In the perl script the DBI module has to be loaded and a connection to the database is established. Then a SQL statement is generated and passed to the database. In the end the connection has to be closed again.

```
#!/usr/bin/perl
use warnings;
use strict;

#load MySQL API
use DBI;
my @con = ( 'DBI:mysql:bercht_db', 'bercht', 'XXX' );
#connect to database
my $dbh = DBI->connect(@con) ||
    die "Database connection not made: $DBI::errstr";
#create table statement
my $sql = "CREATE TABLE disorder (Id VARCHAR(12) NOT
    NULL, Residue LONGTEXT NOT NULL, MD LONGTEXT NOT
    NULL, time DATETIME NOT NULL, PRIMARY KEY id (Id))";
my $sth = $dbh->prepare($sql);
$sth->execute();
$sth->finish();
#close connection
$dbh->disconnect();
```

## 5 Backup

Backups should be made regularly in order to be able to prevent data loss. There are several ways to make a backup of a MySQL database. One option is to use the program *mysqldump* which is included in the mysql-server installation.

```
mysqldump -u root -p -all-databases > dump.sql
```

Other possibilities to do backups are to write a whole table into a delimited-text file:

```
SELECT * INTO OUTFILE 'filename' FROM tbl_name;
```

Furthermore, some storage engines represent each table in a file, so that you can backup them by just copying the corresponding files. For example MyISAM tables can be backed up by copying the corresponding \*.frm, \*.MYD and \*.MYI files.

To test a table for errors the CHECK TABLE command can be executed and to repair them there exists the REPAIR TABLE command [5].

## 6 phpMyAdmin

In order to provide a user-friendly, web-based interface we install the package *phpmyadmin* with aptitude. During the installation process we have to choose apache2 as webserver. Afterwards, we have to include the apache configuration of phpmyadmin in the configuration of our apache server. This can be done by creating a symbolic link to the configuration provided by phpmyadmin in the apache configuration folder (/etc/apache2/conf.d):

```
ln -s /etc/phpmyadmin/apache.conf
```

In this file an alias to the path of phpmyadmin is created and several options are set. If we then reload the apache server and point our browser to "http://localhost/phpmyadmin" we have a user-friendly interface to our MySQL server [6].

## References

- [1] <http://en.wikipedia.org/wiki/database>.
- [2] [http://en.wikipedia.org/wiki/Relational\\_model](http://en.wikipedia.org/wiki/Relational_model).
- [3] <http://en.wikipedia.org/wiki/Sql>.
- [4] <http://en.wikipedia.org/wiki/Mysql>.
- [5] <http://dev.mysql.com/doc/refman/5.1/en/index.html>.
- [6] [http://www.phpmyadmin.net/home\\_page/index.php](http://www.phpmyadmin.net/home_page/index.php).